Postel

Network Working Group
RFC #672
NIC #31440
December 6,1974

RECEIVED
JAN 02 1975

A Multi-Site Data Collection Facility

Richard Schantz

BBN-TENEX

Preface:

This RFC reproduces most of a working document prepared during the design and implementation of the protocols for the TIP-TENEX integrated system for handling TIP accounting. Bernie Cosell (BBN-TIP) and Bob Thomas (BBN-TENEX) have contributed to various aspects of this work. The system has been partially operational for about a month on selected hosts. We feel that the techniques described here have wide applicability beyond TIP accounting.

Section I

Protocols for a Multi-site Data Collection Facility

Introduction

The developement of computer networks has provided the groundwork for distributed computation: one in which a job or task is comprised of components from various computer systems. In a single computer system, the unavailability or malfunction of any of the job components (e.g. program, file, device, etc.) usually necessitates job termination. With computer networks, it becomes feasible to duplicate certain job components which previously had no basis for duplication. (In a single system, it does not matter how many times a process that performs a certain function is duplicated; a system crash makes all unavailable). It is such resource duplication that enables us to utilize the network to achieve high reliability and load leveling. In order to realize the potential of resource duplication, it is necessary to have protocols which provide for the orderly use of these resources. In this document, we first discuss in general terms a problem of protocol definition for interacting with a multiply defined resource (server). The problem deals with providing a highly reliable data collection facility, by supporting it at many sites throughout the network. In the second section of this document, we describe in detail a particular implementation of the protocol which handles the problem of utilizing multiple data collector processes for collecting accounting data generated by the network TIPs. This example also illustrates the specialization of hosts to perform parts of a computation they are best equipped to handle. The large network hosts (TENEX systems) perform the accounting function for the small network access TIPs.

The situation to be discussed is the following: a data generating process needs to use a data collection service which is duplicately provided by processes on a number of network machines. A request to a server involves sending the data to be collected.

An Initial Approach

The data generator could proceed by selecting a particular server and sending its request to that server. It might also take the attitude that if the message reaches the destination host (the communication subsystem will indicate this) the message will be properly processed to completion. Failure of the request message would then lead to selecting another server, until the request succeeds or all servers have been tried.

Such a simple strategy is a poor one. It makes sense to require that the servicing process send a positive acknowledgement to the requesting process. If nothing else, the reply indicates that the server process itself is still functioning. Waiting for such a reply also implies that there is a strategy for selecting another server if the reply is not forthcoming. Herein lies a problem. If the expected reply is timed out, and then a new request is sent to another server, we run the risk of receiving the (delayed) original acknowledgement at a later time. This could result in having the data entered into the collection system twice (data duplication). If the request is re-transmitted to the same server only, we face the possibility of not being able to access a collector (data loss). In addition, for load leveling purposes, we may wish to send new requests to some (or all) servers. We can then use their reply (or lack of reply) as an indicator of load on that particular instance of the service. Doing this without data duplication requires more than a simple request and acknowledgement protocol*.

Extension of the Protocol

The general protocol developed to handle multiple collection servers involves having the data generator send the data request to some (or all) data collectors. Those willing to handle the request reply with an "I´ve got it" message. They then await further notification before finalizing the processing of the data. The data generator sends a "go ahead" message to one of the replying collectors, and a "discard" message to all other replying collectors. The "go ahead" message is the signal to process the data (i.e. collect permanently), while the "discard" message indicates that the data is being collected elsewhere and should not be retained.

The question now arises as to whether or not the collector process should acknowledge receipt of the "go ahead" message with a reply of its own, and then should the generator process acknowledge this acknowledgement, etc. We would like to send as few messages as possible to achieve reliable communication. Therefore, when a state

---

* If the servers are independent of each other to the extent that if two or more servers all act on the same request, the end result is the same as having a single server act on the request, then a simple request/acknowledgement protocol is adequate. Such may be the case, for example, if we subject the totality of collected data (i.e. all data collected by all collectors for a certain period) to a duplicate detection scan. If we could store enough context in each entry to be able to determine duplicates, then having two or more servers act on the data would be functionally equivalent to processing by a single server.

is reached for which further acknowledgements lead to a previously visited state, or when the cost of further acknowledgements outweigh the increase in reliability they bring, further acknowledgements become unnecessary.

The initial question was should the collector process acknowledge the "go ahead" message? Assume for the moment that it should not send such an acknowledgement. The data generator could verify, through the communication subsystem, the transmission of the "go-ahead" message to the host of the collector. If this message did not arrive correctly, the generator has the option of re-transmitting it or sending a "go ahead" to another collector which has acknowledged receipt of the data. Either strategy involves no risk of duplication. If the "go ahead" message arrives correctly, and a collector acknowledgement to the "go ahead" message is not required, then we incur a vulnerability to (collector host) system crash from the time the "go ahead" message is accepted by the host until the time the data is totally processed. Call the data processing time P. Once the data generator has selected a particular collector (on the basis of receiving its "I´ve got it" message), we also incur a vulnerability to malfunction of this collector process. The vulnerable period is from the time the collector sends its "I´ve got it" message until the time the data is processed. This amounts to two network transit times (2N) plus IMP and host overhead for message delivery (O) plus data processing time (P). [Total time=2N+P+O]. A malfunction (crash) in this period can cause the loss of data. There is no potential for duplication.

Now, assume that the data collector process must acknowledge the "go ahead" message. The question then arises as to when such an acknowledgement should be sent. The reasonable choices are either immediately before final processing of the data (i.e. before the data is permanently recorded) or immediately after final processing. It can be argued that unless another acknowledgement is required (by the generator to the collector) to this acknowledgement BEFORE the actual data update, then the best time for the collector to acknowledge the "go ahead" is after final processing. This is so because receiving the acknowledgement conveys more information if it is sent after processing, while not receiving it (timeout), in either case, leaves us in an unknown state with respect to the data update. Depending on the relative speeds of various network and system components, the data may or may not be permanently entered. Therefore if we interpret the timeout as a signal to have the data processed at another site, we run the risk of duplication of data. To avoid data duplication, the timeout strategy must only involve re-sending the "go ahead" message to the same collector. This will only help if the lack of reply is due to a lost network message. Our vulnerability intervals to system and process malfunction remain as before.

It is our conjecture (to be analyzed further) that any further acknowledgements to these acknowledgements will have virtually no effect on reducing the period of vulnerability outlined above. As

such, the protocol with the fewest messages required is superior.

Data Dependent Aspects of the Protocol

As discussed above, a main issue is which process should be the last to respond (send an acknowledgement). If the data generator sends the last message (i.e. "go ahead"), we can only check on its correct arrival at the destination host. We must "take on faith" the ability of the collector to correctly complete the transaction. This strategy is geared toward avoiding data duplication. If on the other hand, the protocol specifies that the collector is to send the last message, with the timeout of such a message causing the data generator to use another collector, then the protocol is geared toward the best efforts of recording the data somewhere, at the expense of possible duplication.

Thus, the nature of the problem will dictate which of the protocols is appropriate for a given situation. The next section deals in the specifics of an implementation of a data collection protocol to handle the problem of collecting TIP accounting data by using the TENEX systems for running the collection server processes. It is shown how the general protocol is optimized for the accounting data collection.

Section II

Protocol for TIP-TENEX Accounting Server Information Exchange

Overview of the Facility

When a user initially requests service from a TIP, the TIP will perform a broadcast ICP to find an available RSEXEC which maintains an authentication data base. The user must then complete a login sequence in order to authenticate himself. If he is successful the RSEXEC will transmit his unique ID code to the TIP. Failure will cause the RSEXEC to close the connection and the TIP to hang up on the user. After the user is authenticated, the TIP will accumulate accounting data for the user session. The data includes a count of messages sent on behalf of the user, and the connect time for the user. From time to time the TIP will transmit intermediate accounting data to Accounting Server (ACTSER) processes scattered throughout the network. These accounting servers will maintain files containing intermediate raw accounting data. The raw accounting data will periodically be collected and sorted to produce an accounting data base. Providing a number of accounting servers reduces the possibility of being unable to find a repository for the

intermediate data, which otherwise would be lost due to buffering limitations in the TIPs. The multitude of accounting servers can also serve to reduce the load on the individual hosts providing this facility.

The rest of this document details the protocol that has been developed to ensure delivery of TIP accounting data to one of the available accounting servers for storage in the intermediate accounting files.

Adapting the Protocol

The TIP to Accounting Server data exchange uses a protocol that allows the TIP to select for data transmission one, some, or all server hosts either sequentially or in parallel, yet insures that the data that becomes part of the accounting file does not contain duplicate information. The protocol also minimizes the amount of data buffering that must be done by the limited capacity TIPs. The protocol is applicable to a wide class of data collection problems which use a number of data generators and collectors. The following describes how the protocol works for TIP accounting.

Each TIP is responsible for maintaining in its memory the cells indicating the connect time and the number of messages sent for each of its current users. These cells are incremented by the TIP for every quantum of connect time and message sent, as the case may be. This is the data generation phase. Periodically, the TIP will scan all its active counters, and along with each user ID code, pack the accumulated data into one network message (i.e. less than 8K bits). The TIP then transmits this data to a set of Accounting Server processes residing throughout the network. The data transfer is over a specially designated host-host link. The accounting servers utilize the raw network message facility of TENEX 1.32 in order to directly access that link. When an ACTSER receives a data message from a TIP, it buffers the data and replies by returning the entire message to the originating TIP. The TIP responds with a positive acknowledgement ("go ahead") to the first ACTSER which returns the data, and responds with a negative acknowledgement ("discard") to all subsequent ACTSER data return messages for this series of transfers. If the TIP does not receive a reply from any ACTSER, it accumulates new data (i.e. the TIP has all the while been incrementing its local counters to reflect the increased connect time and message count; the current values will comprise new data transfers) and sends the new data to the Accounting Server processes. When an ACTSER receives a positive acknowledgement from a TIP (i.e. "go ahead"), it appends the appropriate parts of the buffered data to the locally maintained accounting information file. On receiving a negative acknowledgement from the TIP (i.e. "discard"), the ACTSER discards the data buffered for this TIP. In addition, when the TIP responds with a "go ahead" to the first ACTSER which has accepted the data (acknowledged by returning the intermediate data, which otherwise would be lost due to buffering limitations in the TIPs. The multiple use of accounting servers can also serve to reduce the load on the individual hosts providing this facility.

The rest of this document details the protocol that has been developed to ensure delivery of TIP accounting data to one of the available accounting servers for storage in the intermediate accounting files.

[illegible] the Protocol

The TIP to Accounting Server data exchange uses a protocol that allows the TIP to select for data transmission any, some, or all server hosts either sequentially or in parallel, yet insures that the data that becomes part of the accounting file does not contain duplicate information. The protocol also minimizes the amount of data buffering that must be done by the limited capacity TIPs. The protocol is applicable to a wide class of data collection problems which use a number of data generators and collectors. The following describes how the protocol works for TIP accounting.

Each TIP is responsible for maintaining in its memory the cells indicating the connect time and the number of messages sent for each of its current users. These cells are incremented by the TIP for every quantum of connect time and message sent, as the case may be. This is the data generation phase. Periodically, the TIP will scan all the active counters, and along with each user ID code, pack the accumulated data into one network message (i.e., less than 8K bits). The TIP then transmits this data to a set of Accounting Server processes residing throughout the network. The data transfer is over a specially designated host-host link. The accounting servers utilize the raw network-message facility of NCP's 1.5? in order to directly access that link. When an ACTSER receives a data message from a TIP, it buffers the data and replies by returning the entire message to the originating TIP. The TIP responds with a positive acknowledgement ("go ahead") to the first ACTSER which returns the data, and responds with a negative acknowledgement ("discard") to all subsequent ACTSER data return messages for this series of transfers. If the TIP does not receive a reply from any ACTSER, it accumulates new data (i.e., the TIP has all the while been incrementing its local counters as before, the increased connect time and message count) the current values will comprise new data transfers, and sends the new data to the Accounting Server processes. When an ACTSER receives a positive acknowledgement from a TIP (i.e., "go ahead"), it appends the appropriate parts of the buffered data to the locally maintained accounting information file. On receiving a negative acknowledgement from the TIP (i.e., "discard"), the ACTSER discards the data buffered for this TIP. In addition, when the TIP responds with a "go ahead" to the first ACTSER which has accepted the data (acknowledged by returning the data along with the "I've got it"), the TIP decrements the connect time and message counters for each user by the amount indicated in the data returned by the ACTSER. This data will already be accounted for in the intermediate accounting files.

As an aid in determining which ACTSER replies are to current requests, and which are tardy replies to old requests, the TIP maintains a sequence number indicator, and appends this number to each data message sent to an ACTSER. On receiving a reply from an ACTSER, the TIP merely checks the returned sequence number to see if this is the first reply to the current set of TIP requests. If the returned sequence number is the same as the current sequence number, then this is the first reply; a positive acknowledgement is sent off, the counters are decremented by the returned data, and the sequence number is incremented. If the returned sequence number is not the same as the current one (i.e. not the one we are now seeking a reply for) then a negative acknowledgement is sent to the replying ACTSER. After a positive acknowledgement to an ACTSER (and the implied incrementing of the sequence number), the TIP can wait for more information to accumulate, and then start transmitting again using the new sequence number.

Further Clarification of the Protocol

There are a number of points concerning the protocol that should be noted.

1. The data generator (TIP) can send different (i.e. updated versions) data to different data collectors (accounting servers) as part of the same logical transmission sequence. This is possible because the TIP does not account for the data sent until it receives the acknowledgement of the data echo. This strategy relieves the TIP of any buffering in conjunction with re-transmission of data which hasn't been acknowledged.

2. A new data request to an accounting server from a TIP will also serve as a negative acknowledgement concerning any data already buffered by the ACTSER for that TIP, but not yet acknowledged. The old data will be discarded, and the new data will be buffered and echoed as an acknowledgement. This allows the TIP the option of not sending a negative acknowledgement when it is not convenient to do so, without having to remember that it must be sent at a later time. There is one exception to this convention. If the new data message has the same sequence number as the old buffered message, then the new data must be discarded, and the old data kept and re-echoed. This is to prevent a slow acknowledgement to the old data from being accepted by the TIP, after the TIP has already sent the new data to the slow host. This caveat can be avoided if the TIP does not resend to a non-responding server within the time period that a message could possibly be stuck in the network, but could still be delivered. Ignoring this situation may result in some accounting

data being counted twice. Because of the rule to keep old data when confronted with matching sequence numbers, on restarting after a crash, the TIP should send a "discard" message to all servers in order to clear any data which has been buffered for it prior to the crash. An alternative to this would be for the TIP to initialize its sequence number from a varying source such as time of day.

3. The accounting server similarly need not acknowledge receipt of data (by echoing) if it finds itself otherwise occupied. This will mean that the ACTSER is not buffering the data, and hence is not a candidate for entering the data into the file. However, the TIP may try this ACTSER at a later time (even with the same data), with no ill effects.

4. Because of 2 and 3 above, the protocol is robust with respect to lost or garbled transmissions of TIP data requests and accounting server echo replies. That is, in the event of loss of such a message, a re-transmission will occur as the normal procedure.

5. There is no synchronization problem with respect to the sequence number used for duplicate detection, since this number is maintained only at the TIP site. The accounting server merely echoes the sequence number it has received as part of the data.

6. There are, however, some constraints on the size of the sequence number field. It must be large enough so that ALL traces of the previous use of a given sequence number are totally removed from the network before the number is re-used by the TIP. The sequence number is modulo the size of the largest number represented by the number of bits allocated, and is cyclic. Problems generally arise when a host proceeds from a service interruption while it was holding on to a reply. If during the service interruption, we have cycled through our sequence numbers exactly N times (where N is any integer), this VERY tardy reply could be mistaken for a reply to the new data, which has the same sequence number (i.e. N revolutions of sequence numbers later). By utilizing a sufficiently large sequence number field (16 bits), and by allowing sufficient time between instances of sending new data, we can effectively reduce the probability of such an error to zero.

7. Since the data involved in this problem is the source of accounting information, care must be taken to avoid duplicate entries. This must be done at the expense of potentially losing data in certain instances. Other than the obvious TIP malfunction, there are two known ways of losing data. One is the situation where no accounting server responds to a TIP for an extended period of time causing the TIP counters to overflow (highly unlikely if there are sufficient Accounting Servers). In this case, the TIP can hold the counters at their maximum value until a server comes up, thereby keeping the lost accounting data at its minimum. The other situation results from adapting the protocol to our insistence on no duplicate data in the incremental files. We are vulnerable to data loss with no recourse from the time the server receives the "go

ahead" to update the file with the buffered data (i.e. positive acknowledgement) until the time the update is completed and the file is closed. An accounting server crash during this period will cause that accounting data to be lost. In our initial implementation, we have slightly extended this period of vulnerability in order to save the TIP from having to buffer the acknowledged data for a short period of time. By updating TIP counters from the returned data in parallel with sending the "go ahead" acknowledgement, we relieve the TIP of the burden of buffering this data until the Request for Next Message (RFNM) from the accounting server IMP is received. This adds slightly to our period of vulnerability to malfunction, moving the beginning of the period from the point when the ACTSER host receives the "go ahead", back to the point when the TIP sends off the "go ahead" (i.e. a period of one network transit time plus some IMP processing time). However, loss of data in this period is detectable through the Host Dead or Incomplete Transmission return in place of the RFNM. We intend to record such occurrences with the Network Control Center. If this data loss becomes intolerable, the TIP program will be modified to await the RFNM for the positive acknowledgement before updating its counters. In such a case, if the RFNM does not come, the TIP can discard the buffered data and re-transmit new data to other servers.

8. There is adequate protection against the entry of forged data into the intermediate accounting files. This is primarily due to the system enforced limited access to Host-Imp messages and Host-Host links. In addition, messages received on such designated limited access links can be easily verified as coming from a TIP. The IMP subnet appends the signature (address) of the sending host to all of its messages, so there can be no forging. The Accounting Server is in a position to check if the source of the message is in fact a TIP data generator.

Current Parameters of the Protocol

In the initial implementation, the TIP sends its accumulated accounting data about once every half hour. If it gets no positive acknowledgement, it tries to send with greater frequency (about every 5 minutes) until it finally succeeds. It can then return to the normal waiting period. (A TIP user logout introduces an exception to this behavior. In order to re-use the TIP port and its associated counters as soon as possible, a user terminating his TIP session causes the accounting data to be sent immediately). Initially, our implementation calls for each TIP to remember a "favored" accounting server. At the wait period expiration, the TIP will try to deposit the data at its "favored" site. If successful within a short timeout period, this site remains the favored site, and the wait interval is reset. If unsuccesful within the short ahead" to update the file with the buffered data (i.e. positive acknowledgement) until the time the update is completed and the file is closed. An accounting server crash during this period will cause that accounting data to be lost. In our initial implementation, we have slightly extended this period of vulnerability in order to save the TIP from having to buffer the large accounting data for a short period of time. By updating TIP accounting from the returned data in parallel with sending the "go ahead" acknowledgement, we relieve the TIP of the burden of buffering this data until the Request For Next Message (RFNM) from the accounting server's IMP is received. This adds slightly to our period of vulnerability to malfunction, moving the beginning of the period from the point when the ACCESS host receives the "go ahead", back to the point when the TIP sends off the "go ahead" (i.e., a period of one network transit time plus some IMP processing time). However, loss of data in this period is detectable through the Host-Host Protocol's "Incomplete Transmission" return in place of the RFNM. We intend to record such occurrences with the Network Control Center. If this data loss becomes intolerable, the TIP program will be modified to await the RFNM for the positive acknowledgement before updating its counters. In such a case, if the RFNM does not come, the TIP can discard the buffered data and re-transmit new data to other servers.

2. There is adequate protection against the entry of forged data into the intermediate accounting files. This is primarily due to the system enforced limited access to host-host messages and host-host links. In addition, messages received on such designated limited access links can be easily verified as coming from a TIP. (The IMP subnet appends the signature (address) of the sending host to all of its messages, so there can be no forging.) The Accounting Server is in a position to check if the sender of the message is in fact a TIP data generator.

Current Parameters of the Protocol

In the initial implementation, the TIP sends its accumulated accounting data about once every half hour. If it gets no positive acknowledgement, it tries to send again, with greater frequency (every 5 minutes) until it finally succeeds. It can then return to the normal waiting period. (A TIP user logout is an exception to this behavior. In order to re-use the TIP port and its associated counters as soon as possible, a user terminating his TIP session causes the accounting data to be sent immediately.) Initially, our implementation calls for each TIP to have a "favored" accounting server. At the wait period expiration, the TIP will try to deposit its data at its "favored" site. If successful within a short timeout period, this site remains the favored site, and the wait interval is reset. If unsuccessful through the short timeout, the data can be sent to all servers*. The one replying first will update its file with the data and also become the "favored" server for this TIP. With these parameters, a host would have to undergo a proceedable service interruption of more than a year in order for the potential sequence number problem outlined in (6) above to occur.

Concluding Remarks

When the implementation is complete, we will have a general data accumulation and collection system which can be used to gather a wide variety of information. The protocol as outlined is geared to gathering data which is either independent of the previously accumulated data items (e.g. recording names), or data which adheres to a commutative relationship (e.g. counting). This is a consequence of the policy of retransmission of different versions of the data to different potential collectors (to relieve TIP buffering problems).

In the specified version of the protocol, care was taken to avoid duplicate data entries, at the cost of possibly losing some data through collector malfunction. Data collection problems which require avoiding such loss (at the cost of possible duplication of some data items) can easily be accommodated with a slight adjustment to the protocol. Collected data which does not adhere to the commutative relationship indicated above, can also be handled by utilizing more buffer space at the data generator sites.

---

* The sequence number can be incremented for this new set of data messages, and the new data can also be sent to the slow host. In this way we won´t be giving the tardy response from the old favored host unfair advantage in determining which server can respond most quickly. If there is no reply to this series of messages, the TIP can continue to resend the new data. However, the sequence number should not be incremented, since no reply was received, and since indiscriminate incrementing of the sequence number increases the chance of recycling during the lifetime of a message.